Package 'gfoRmulaICE'

February 14, 2024

Type Package

Title ICE

Version 0.1.0

Author

Maintainer

Description Implements the parametric g-formula iterative conditional estimator (ICE) algorithms.
The ICE estimator can be used to estimate the causal effects of time-varying
treatment interventions on risk of a survival outcome from longitudinal data with
time-varying confounding.
The package supports 1) both singly robust and doubly robust estimators;
2) static, dynamic, deterministic, stochastic, and threshold interventions; 3) censoring event;
4) competing event; 5) user-defined intervention strategies.

License

Encoding UTF-8

LazyData true

Imports data.table,
ggplot2,
nnet,
parallel,
doParallel,
foreach,
doRNG,
stats,
stringr,
survival,
tidyverse

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.2.2

R topics documented:

`bootstrap_ice` *Bootstrap for ICE estimator*

Description

This function estimates the variance of the point estimates for all user-defined interventions from the ICE estimator object. The variance is calculated by bootstrapping and is used to calculate the confidence interval for each point estimate of each user-defined intervention specified in the ICE estimator object.

Usage

```
bootstrap_ice(
  f,
  K,
  nboot,
  sig_level,
  parallel,
  ncores,
  ref_description,
  ref_intervention_varname,
  ref_total_effect,
  total_effect,
  natural_course_type,
  ref_intervention,
  interventions,
  intervention_varnames,
  intervention_description,
  data,
  id,
  set_seed,
  ...
)
```

Arguments

f	a function specifying which ICE estimator to use for bootstrap.
K	a numerical value that indicates the total number of time points.
nboot	a numeric indicating number of bootstrap samples.
sig_level	a numeric indicating the significance level to be used for confidence interval.
parallel	a logical indicating whether to parallelize the bootstrap process.
ncores	a numeric indicating the number of CPU cores to be used in parallel computing.
ref_description	a string specifying a description for the reference intervention.
ref_intervention_varname	a string specifying the intervention variable used in the reference intervention.
ref_total_effect	a logical indicating how the competing event is handled for the reference intervention. TRUE for total effect. FALSE for direct effect.
total_effect	a logical indicating how the competing event is handled for the defined intervention. TRUE for total effect. FALSE for direct effect.
natural_course_type	a character string indicating how the natural course strategy is computed for the defined intervention. "ipw" for IPW natural course risk estimation.
ref_intervention	a list of function specifying the intervention to be used as reference. for the inverse probability weighted estimate of the natural course risk in the reference intervention.
interventions	a list of functions defining the interventions to be used in this bootstrap.
intervention_varnames	a list of character strings corresponding the interventions to be used in this bootstrap.
intervention_description	a character string specifying a description for the define intervention in this bootstrap.
data	a data frame containing the observed data in long format.
id	a character string indicating the ID variable name in data.
set_seed	a numeric indicating the starting seed for bootstrap.
...	any keyword arguments to be passed in f.

Value

A list containing the following components:

ice_se	The standard error for the ICE risk of the defined intervention.
ref_se	The standard error for the ICE risk of the reference intervention.
rr_se	The standard error for the ICE risk of the risk ratio.
rd_se	The standard error for the ICE risk of the risk difference.
ice_cv	The empirical quantile for the ICE risk of the defined intervention.
ref_cv	The empirical quantile for the ICE risk of the reference intervention.
rr_cv	The empirical quantile for the ICE risk of the risk ratio.

rd_cv	The empirical quantile for the ICE risk of the risk difference.
boot_data	A list containing all the bootstrapped data samples.
boot_models	A list containing the fitted models for the outcome, the treatment (if applicable), and the competing event (if applicable) on the bootstrapped samples.
boot_summary	A list containing the summary of the fitted models on the bootstrapped samples.
boot_stderr	A list containing the standard errors of the coefficients of the fitted models on the bootstrapped samples.
boot_vcov	A list containing the variance-covariance matrices of the parameters of the fitted models on the bootstrapped samples.
boot_rmse	A list containing the root mean square error (RMSE) values of the fitted models on the bootstrapped samples.

`compute_weighted_hazard` *Calculate Observed Natural Course Risk*

Description

This functions calculates the inverse probability weighted observed natural course risk.

Usage

```
compute_weighted_hazard(
  prob_censor,
  data,
  id,
  censor_varname,
  time_points,
  time_name,
  outcome_name,
  competing_varname,
  competing_fit,
  total_effect
)
```

Arguments

prob_censor	a vector of numerics specifying the estimated probability of censoring for each individual.
data	a data frame containing the observed data.
id	a character string indicating the ID variable name in data.
censor_varname	a character string indicating the censor variable name in data.
time_points	a numeric that indicates the total number of time points.
time_name	a character string indicating the time variable name in data.
outcome_name	a character string indicating the outcome variable name in data.
competing_varname	a character string indicating the competing event variable name in data.
competing_fit	a fitted model for the competing event model.
total_effect	a logical indicating whether to treat competing event as censoring or total effect.

Value

A list with the first entry as a vector of the mean observed risk. Its second entry is a vector of mean observed survival. Its third entry is a vector of inverse probability weight.

`comp_and_censor_data` *Example Dataset for a Survival Outcome with Both Censoring and Competing Event*

Description

A dataset with 26581 observations on 10000 individuals and 4 time points. The dataset is in long format with each row representing the record of one individual at one time point.

Usage

```
data
```

Format

A data frame with 26581 rows and 9 variables:

t0 Time index.

id Unique identifier for each individual.

L1 Binary covariate.

L2 Continuous covariate.

A1 Categorical treatment variable with levels 1, 2, and 3.

A2 Binary treatment variable.

C Censoring event indicator.

D Competing event indicator, time-varying indicator of failure.

Y Survival outcome, time-varying indicator of failure.

`dynamic` *Dynamic Intervention Function*

Description

This function implements the dynamic intervention strategy. Dynamic intervention strategy refers to the class of treatment mechanisms that depend on the covariate values. This function provides an example of dynamic intervention strategy. The specific mechanism is described as following:

- For `type` being `compare`, the treatment is determined based on the condition of `direction` and `value`. For example, if `direction` is >, then treat if `var` > `value`, and not treat otherwise.
- For `type` being `absorbing`, the initiation of treatment is based on the same mechanism as `type` being `compare`, and adopts always treat once the treatment is initiated for each individual.

Possible `direction` includes `>`,`<`,`>=`,`<=`,`=`,`!=`.

Usage

```
dynamic(type, var, direction, value, id = id_var, data = interv_data)
```

Arguments

type	a character string specifying the type of dynamic intervention. Possible inputs are compare and absorbing.
var	a character string specifying the intervention variable name in data.
direction	a character string specifying the comparison of var and value. Possible inputs are >,<,>=,<=,=,!=.
value	a numeric specifying the comparison value on var.
id	a character string specifying the id variable in data.
data	a data frame containing the observed data.

Value

a vector containing the intervened value in the same size as the number of rows in data.

Examples

```
data <- readRDS("test_data_competing.rds")
dynamic <- dynamic(type = "absorbing", var = "L1", value = 0, direction = ">", id = "id", data = data)
dynamic
```

---

grace_period *Intervention with Grace Period Strategy*

Description

This function implements a particular intervention with grace period. There are two choices of grace type: "natural" or "uniform," specified by the argument type.

- Natural grace period describes the following mechanism: When the condition var = value is met, initiate treatment in n time units, where n is specified by nperiod. If there is no intervention, follow the observed treatment initiation distribution.
- Uniform grace period describes the following mechanism: When the condition var = value is met, initiate treatment in n time units, where n is specified by nperiod. If there is no intervention, follow the uniform distribution of treatment initiation.

Usage

```
grace_period(
  type,
  nperiod,
  var,
  value,
  data = interv_data,
  id = idvar,
  time_name = time0var,
  outcome_name = outcomevar
)
```

Arguments

type	a character string specifying the type of grace period strategy. "Uniform" for uniform grace period, and "natural" for natural grace period.
nperiod	a numeric specifying the length of grace period.
var	a character string specifying the variable that the grace period strategy is based on.
value	a numeric specifying the value that var is compared to.
data	a data frame containing the observed data.
id	a character string indicating the ID variable name in data.
time_name	a character string indicating the time variable name in data.
outcome_name	a character string indicating the outcome variable name in data.

Value

a vector containing the intervened value in the same size as the number of rows in data.

Examples

```
data <- readRDS("test_data_competing.rds")
grace_period <- grace_period(type = "uniform", nperiod = 2, var = "L1", value = 1, data = data,
                             id = "id", time_name = "t0", outcome_name = "Y")
grace_period
```

---

ice *Singly Robust and Doubly Robust Iterative Conditional Expectation (ICE) Estimator for the Specified Intervention*

Description

This function estimates the risk over time for survival outcome using the given observed data set following multiple user-defined intervention strategies by the parametric g-formula iterative conditional expectation (ICE) estimator. This function allows users to access all singly robust and doubly robust ICE estimators: classical pooling over treatment history ICE estimator, classical stratifying on treatment history ICE estimator, hazard-based pooling over treatment history ICE estimator, hazard-based stratifying on treatment history ICE estimator, and a doubly robust inverse probability weighted ICE estimator. Please see Wen et al. (2021) more details regarding the parameter g-formula iterative conditional expectation estimator.

Usage

```
ice(
  data,
  time_points,
  id,
  time_name,
  outcome_name,
  censor_name = NULL,
  compevent_name = NULL,
  comp_effect = 0,
```

```
  outcome_model,
  censor_model = NULL,
  competing_model = NULL,
  ref_idx = 0,
  estimator,
  int_descript,
  ci_method = "percentile",
  nsamples = 0,
  seed = 1,
  significance_level = 0.05,
  parallel = F,
  ncores = 2,
  ...
)
```

Arguments

data	a data frame containing the observed data in long format.
time_points	a numerical value that indicates the total number of time points.
id	a character string indicating the ID variable name in data.
time_name	a character string indicating the time variable name in data.
outcome_name	a character string indicating the outcome variable name in data.
censor_name	a character string indicating the censor variable name in data. Default is NULL.
compevent_name	a character string indicating the competing variable name in data. Default is NULL.
comp_effect	a numeric specifying how the competing event is handled for all the specified interventions. Default is 0. 0 outputs direct effect for all the specified interventions. 1 outputs total effect for all the specified interventions.
outcome_model	a formula specifying the model statement for the outcome model.
censor_model	a formula specifying the model statement for the censoring model for IP weighted natural course risk. Default is NULL.
competing_model	a formula specifying the model statement for the competing model for hazard-based ICE estimator. Default is NULL.
ref_idx	a numerical indicating which intervention to be used as the reference to calculate the risk ratio and risk difference. Default is 0. 0 refers to using the natural course as the reference intervention. Any other numbers refer to the corresponding intervention that users specify in the keywords argument.
estimator	a function to specifying which ICE estimator to use for the estimation. Possible inputs are: • Classical Pooling over Treatment History ICE Estimator: pool(hazard = F) • Hazard Based Pooling over Treatment History ICE Estimator: pool(hazard = T) • Classical Stratifying Treatment History ICE Estimator: strat(hazard = F) • Hazard Based Stratifying Treatment History ICE Estimator: strat(hazard = T)

- Doubly Robust Weighted ICE Estimator:
  `weight(treat_model)` where `treat_model` is a list specifying the treatment model.

`int_descript` a vector of strings containing description for each specified intervention.

`ci_method` a character string specifying the method used for calculating the confidence interval, if `nsamples` is larger than 0. Either "percentile" or "normal." Default is "percentile."

`nsamples` a numeric indicating number of bootstrap samples. If a number larger than 0 is specified, bootstrap samples will be used for standard error estimate and confidence interval. Default is 0.

`seed` a numeric indicating the starting seed for bootstrap. Default is 1.

`significance_level`
a numeric indicating the significance level to be used for confidence interval. Default is 0.05.

`parallel` a logical indicating whether to parallelize the bootstrap process. TRUE for using parallel computing. FALSE for not using parallel computing. Default is FALSE.

`ncores` a numeric indicating the number of CPU cores to be used in parallel computing. Default is 2.

`...` keywords arguments for intervention inputs, optional outcome models for stratified ICE, and optional competing models for stratified ICE. The keyword argument for interventions should follow the below naming convention:

- Each intervention is specified using the keyword argument name with *intervention* prefix.
- Use *i* after *intervention* prefix in keyword argument name to represent the ith intervention strategy.
- Use . followed with *treatment variable name* after *interventioni* in keyword argument name to represent the treatment name within the ith intervention strategy.

The input for each keyword argument must be a list encompassing two elements, which are: a vector of intervened values and an optional vector of time points on which the corresponding intervention is applied. If the second element is not specified, then the defined intervention will be applied to all time points. A sample intervention input with two treatments of names A1 and A2 and two intervention strategies look like:

```
intervention1.A1 = list(static(1))
intervention1.A2 = list(dynamic("compare","L1",">",0))
intervention2.A1 = list(static(0))
intervention2.A2 = list(dynamic("absorbing","L1",">",1))
```

A sample intervention input with one treatment of name A and two intervention strategies look like:

```
intervention1.A1 = list(static(1))
intervention2.A1 = list(static(0))
```

For the above two intervention inputs, since there is no second element input, each specified intervention is applied on all time points. If one wants to specify the custom time points on which each intervention is applied, a sample intervention input looks like: `intervention1.A1 = list(static(1),1:2)`
`intervention1.A2 = list(static(0),3:5)`
where the intervention on A1 within intervention strategy 1 is applied on time 1 and 2, and the intervention on A2 within intervention strategy 1 is applied on

time 3 to 5. If there is no intervention specified, only the natural course risk will be returned.
To specify different outcome model for different intervention, please follow the keyword argument input convention below:

- Each outcome model is specified using keyword argument name starting with *outcomeModel* prefix.
- Use *.n outcomeModel* prefix in keyword argument name to specify which intervention being applied to, where *n* represents the *n*th intervention.

To specify different competing model for different intervention, please follow the keyword argument input convention below:

- Each competing model is specified using keyword argument name starting with *compModel* prefix.
- Use *.n compModel* prefix in keyword argument name to represent applying to *n*th intervention.

The input to each outcome or competing model keyword argument is a model statement formula. Please refer to the examples section for more examples.

Details

Users could specify which version ICE estimator to use through `estimator`.

- `pool(hazard = F)` specifies the classical pooling over treatment history ICE estimator.
- `pool(hazard = T)` specifies the hazard-based pooling over treatment history ICE estimator.
- `strat(hazard = F)` specifies the classical stratifying on treatment history ICE estimator.
- `strat(hazard = T)` specifies the hazard-based stratify on treatment history ICE estimator.
- `weight(treat_model)` specifies the IP weighted ICE estimator where `treat_model` specifies the treatment model.

For stratified ICE, the estimator will automatically add in the treatment variables that are not intervened as covariates into the model specification for each time point. To provide flexible choices on the model specification inputs, we also provide keyword argument options for users to specify a different model statement, which is different from the model statement specified in `outcome_model` or `competing_model`, for user-chosen interventions. The input to each outcome or competing model keyword argument is a model statement formula. To specify different outcome model for different intervention, please follow the keyword argument input convention below:

- Each outcome model is specified using keyword argument name starting with *outcomeModel* prefix.
- Use *.n outcomeModel* prefix in keyword argument name to specify which intervention being applied to, where *n* represents the *n*th intervention.

To specify different competing model for different intervention, please follow the keyword argument input convention below:

- Each competing model is specified using keyword argument name starting with *compModel* prefix.
- Use *.n compModel* prefix in keyword argument name to represent applying to *n*th intervention.

If an outcome or competing model is specified for an intervention using keyword argument, then the outcome or competing model will be used for the intervention. If there is no outcome or competing model keyword argument specified, the outcome or competing model specified in `outcome_model` or `competing_model` will be used for the intervention.

For example, the following input specifies `Y ~ L1` as outcome model for intervention 1, `D ~ L1` as competing model for intervention 2, `D ~ L1 + L2` as competing model for intervention 1, `Y ~ L1 + L2` as outcome model for intervention 2.

```
fit_hazard_strat <-ice(data = data,K = 4,id = "id",time_name = "t0",
outcome_name = "Y",censor_name = "C",competing_name = "D",
estimator = strat(hazard = T),comp_effect = 1,
censor_model = C ~ L1 + L2,ref_idx = 0,
int_descript = c("Static Intervention","Dynamic Intervention"),
outcome_model = Y ~ L1 + L2,
competing_model = D ~ L1 + L2,
intervention1.A1 = list(static(0)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic("absorbing","L1","=",1)),
intervention2.A2 = list(dynamic("compare","L1","=",0)),
outcomeModel.1 = Y ~ L1,
compModel.2 = D ~ L1
```

Because the keyword argument for outcome model is not specified for intervention 2, the outcome model for intervention 2 is `Y ~ L1 + L2` as specified in `outcome_model`. Similarly, because the keyword argument for competing model is not specified for intervention 1, the competing model for intervention 1 is `D ~ L1 + L2` as specified in `competing_model`. Please see more examples in the examples section.

Users could specify user-defined interventions or implement built-in interventions provided by the package using the intervention input convention described in the parameter description section.

For the package built-in intervention strategy functions, if one wants to implement the strategy within the ICE method, please follow the instructions described below. Furthermore, users could assess the intervened values by specifying additional arguments following the detailed documentation for each function. Built-in interventions include following:

- Always Treat: `static(1)`
- Never Treat: `static(0)`
- Dynamic Treat Based on `var`: `dynamic(type,var,direction,value)`. `type` could be "compare" or "absorbing" and `direction` could be `>,>=,<,<=,=,!=`. If `type` is "compare", then treat if `var` > or >= or < or <= or = or != `value` and not treat otherwise. If `type` is "absorbing", then implements always treat if treatment is initiated based on the description of the "compare" type.
- Threshold Intervention Based on `var`: `threshold(lower_bound,upper_bound,var)`. If treatment value is within the range between `lower_bound` and `upper_bound` inclusively, then follow the natural value of the treatment. Otherwise, if the treatment value is below `lower_bound`, then set to `lower_bound`. If the treatment value is above the upper bound, then set to `upper_bound`.
- Grace Period: `grace_period(type,nperiod,var,value)`. `type` could be "uniform" or "natural". `nperiod` specifies the length of grace period. When a defined intervention condition based on `var = value` is met, initiate treatment in `nperiod` time units. If there is no intervention, follow the observed treatment initiation distribution (for natural grace period) or the uniform distribution of treatment initiation (for uniform grace period).

- User-Defined Intervention: The output of the user-defined intervention should be a vector of intervened value of the intervention variable for each individual at each time point in the size as the number of rows in `data`.

Some examples are provided in the example section.

In order to obtain an inverse probability (IP) weighted natural course risk based on the observed data, users must specify a censoring variable through `censor_name` and a corresponding censoring model through `censor_model`. Please see Chiu et al. (2023) for more details regarding the IP weighted estimate of the natural course risk.

If competing event exists in the data, users need to specify the name of the competing variable through `competing_name` and the model specification through `competing_model` for hazard-based ICE estimator. Users need to specify whether to treat the competing event as censoring or total effect through `total_effect`.

Value

A list containing the following components:

`summary`	A summary table containing the estimated ICE risk, risk ratio, risk difference. If `bootstrap` is TRUE, then the table also includes standard error and confidence interval for ICE risk, risk ratio, and risk difference of each intervention.
`risk.over.time`	A data frame containing the estimated ICE risk at each time point for each intervention.
`models`	A list containing sublists whose names are the specified intervention descriptions, and each sublist contains the fitted models for the outcome, the treatment (if applicable), and the competing event (if applicable) of the corresponding intervention.
`model.summary`	A list containing sublists whose names are the specified intervention descriptions, and each sublist contains the summary of the fitted models for the corresponding intervention.
`model.stderr`	A list containing sublists whose names are the specified intervention descriptions, and each sublist contains the standard errors of the coefficients of the fitted models for the corresponding intervention.
`model.vcov`	A list containing sublists whose names are the specified intervention descriptions, and each sublist contains the variance-covariance matrices of the parameters of the fitted models for the corresponding intervention.
`model.rmse`	A list containing sublists whose names are the specified intervention descriptions, and each sublist contains root mean square error (RMSE) values of the fitted models for the corresponding intervention.
`boot.data`	A list containing all the bootstrapped data if `boostrap` is set to `TRUE`.
`boot.models`	A list containing the fitted models for the outcome, the treatment (if applicable), and the competing event (if applicable) on the bootstrapped samples.
`boot.summary`	A list containing the summary of the fitted models on the bootstrapped samples.
`boot.stderr`	A list containing the standard errors of the coefficients of the fitted models on the bootstrapped samples.
`boot.vcov`	A list containing the variance-covariance matrices of the parameters of the fitted models on the bootstrapped samples.
`boot.rmse`	A list containing the root mean square error (RMSE) values of the fitted models on the bootstrapped samples.
`estimator.type`	A string specifying the type of ICE estimator.

References

Wen L, Young JG, Robins JM, Hernán MA. Parametric g-formula implementations for causal survival analyses. Biometrics. 2021;77(2):740-753.

McGrath S, Lin V, Zhang Z, Petito LC, Logan RW, Hernán MA, and JG Young. gfoRmula: An R package for estimating the effects of sustained treatment strategies via the parametric g-formula. Patterns. 2020;1:100008.

Young JG, Herńan MA, Robins JM. Identification, estimation and approximation of risk under interventions that depend on the natural value of treatment using observational data. Epidemiologic Methods. 2014;3(1):1-19.

Young JG, Vatsa R, Murray EJ, Hernán MA. Interval-cohort designs and bias in the estimation of per-protocol effects: a simulation study. Trials. 2019;20(1):552.

Díaz, I, Williams, N, Hoffman, KL, & Schenck, EJ. Nonparametric causal effects based on longitudinal modified treatment policies. Journal of the American Statistical Association. 2021;118(542), 846–857.

Young JG, Stensrud MJ, Tchetgen Tchetgen EJ, Hernán MA. A causal framework for classical statistical estimands in failure-time settings with competing events. Statistics in medicine. 2020;39(8):1199-1236.

Wen L, Hernán MA, Robins JM. Multiply robust estimators of causal effects for survival outcomes. Scandinavian journal of statistics, theory and applications. 2022;49(3):1304-1328.

Haneuse S, Rotnitzky A. Estimation of the effect of interventions that modify the received treatment. Statistics in medicine. 2013;32(30):5260-5277.

McGrath S, Young JG, Hernán MA. Revisiting the g-null Paradox. Epidemiology. 2022;33(1):114-120.

Chiu YH, Wen L, McGrath S, Logan R, Dahabreh IJ, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. American Journal of Epidemiology. 2023;192:1887–1895.

Examples

```
dynamic_cat <- case_when(comp_and_censor_data$L2 < 0 ~ 1,
comp_and_censor_data$L2 >= 0 & comp_and_censor_data$L2 < 2 ~ 2,
T ~ 3)

# For the following examples, we consider two interventions.
# Intervention 1 is a static intervention, and
# intervention 2 is a dynamic intervention, where the intervention
# for A1 is user-defined.

# Hazard based stratified ICE,
# competing event as total effect for all interventions,
# with Y ~ L1 + L2 as outcome model for both intervention 1 and intervention 2,
# and D ~ L1 + L2 as competing model for both intervention 1 and intervention 2,
# bootstrap with 1000 samples with normal quantile,
# natural course as the reference intervention

ice_strat_haz <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
outcome_model = Y ~ L1 + L2, censor_model = C ~ L1 + L2,
competing_model = D ~ L1 + L2,
comp_effect = 1,
```

```
ref_idx = 0,
estimator = strat(hazard = T),
nsamples = 1000, ci_method = "normal",
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("compare", "L1", "=", 0))
)

plot_risk(ice_strat_haz)

# Hazard based stratified ICE,
# competing event as total effect for all interventions,
# with Y ~ L1 + L2 as outcome model for intervention 1,
# Y ~ L1 as outcome model for intervention 2,
# D ~ L1 as competing model for intervention 1,
# and D ~ L1 + L2 as competing model for intervention 2,
# bootstrap with 1000 samples with normal quantile,
# natural course as the reference intervention

ice_strat_haz <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
outcome_model = Y ~ L1, censor_model = C ~ L1,
competing_model = D ~ L1,
comp_effect = 1,
ref_idx = 0,
estimator = strat(hazard = T),
nsamples = 1000, ci_method = "normal",
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("compare", "L1", "=", 0)),
outcomeModel.1 = Y ~ L1 + L2,
compModel.2 = D ~ L1 + L2
)

plot_risk(ice_strat_haz)

# Classical pooled ICE,
# competing event as direct effect for all interventions,
# bootstrap with 1000 samples using empirical quantile,
# natural course as the reference intervention.

ice_pool_classic <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + A1 + A2, censor_model = C ~ L1 + A1 + A2,
competing_model = D ~ L1 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
```

```
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("compare", "L1", "=", 0))
)

plot_risk(ice_pool_classic)

# Hazard based pooled ICE,
# competing event as direct effect for all interventions,
# bootstrap with 1000 samples using empirical quantile,
# always treat as the reference intervention.

ice_pool_haz <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + A1 + A2, censor_model = C ~ L1 + A1 + A2,
competing_model = D ~ L1 + A1 + A2,
ref_idx = 1,
estimator = pool(hazard = T),
nsamples = 1000, ci_method = "percentile",
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("compare", "L1", "=", 0))
)

plot_risk(ice_pool_haz)

# Doubly robust ICE,
# competing event as direct effect for all interventions,
# with Y ~ L1 as outcome model for both intervention 1 and intervention 2,
# and D ~ L1 as competing model for both intervention 1 and intervention 2,
# bootstrap with 1000 samples using normal quantile,
# natural course as the reference intervention.

ice_weight <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1, censor_model = C ~ L1,
competing_model = D ~ L1,
ref_idx = 0,
estimator = weight(list(A1 ~ L1 + L2, A2 ~ L1 + L2)),
nsamples = 1000, ci_method = "normal",
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("compare", "L1", "=", 0))
)
```

```
plot_risk(ice_weight)

# Doubly robust ICE,
# competing event as direct effect for all interventions,
# with Y ~ L1 as outcome model for intervention 1,
# Y ~ L1 + L2 as outcome model for intervention 2,
# D ~ L1 + L2 as competing model for intervention 1,
# D ~ L1 as competing model for intervention 2,
# bootstrap with 1000 samples using normal quantile,
# natural course as the reference intervention.

ice_weight <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1, censor_model = C ~ L1,
competing_model = D ~ L1,
ref_idx = 0,
estimator = weight(list(A1 ~ L1 + L2, A2 ~ L1 + L2)),
nsamples = 1000, ci_method = "normal",
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("compare", "L1", "=", 0)),
outcomeModel.2 = Y ~ L1 + L2,
compModel.1 = D ~ L1 + L2
)

plot_risk(ice_weight)


# The following example implements the natural value intervention on L1
# (i.e. if L1 > 0, then replace the observed value of L1 by 0,
# and keep the observed value of L1 otherwise),
# dynamic intervention on L1 (treat when L1 = 0)
# with uniform grace period of 2 periods, and
# threshold intervention when the natural value of L2 at time t is lower
# than -3, set its value to -3. Otherwise, do not intervene.

# Classical pooled ICE,
# competing event as direct effect for all interventions,
# bootstrap with 1000 samples using empirical quantile,
# natural course as the reference intervention.

ice_pool_grace_period <- ice(data = comp_and_censor_data, time_points = 4, id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2, censor_model = C ~ L1 + L2 + A1 + A2,
competing_model = D ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
int_descript = c("Grace Period", "Threshold Intervention")
```

```
intervention1.A2 = list(grace_period("uniform", 2, "L1", 0)),
intervention2.L2 = list(threshold(-3))
)

plot_risk(ice_pool_grace_period)
```

`natural_course` *Natural Course Intervention Function*

Description

This function implements the natural course intervention under the observed treatment. This function is to be used in the main ICE estimator as an input in the `interventions` argument.

Usage

```
natural_course(data = interv_data, treat_var = treatment_varname)
```

Arguments

`data`	a data frame containing the observed data.
`treat_var`	a character string specifying the observed treatment variable in `data`.

Value

a vector containing the intervened value in the same size as the number of rows in `data`.

Examples

```
data <- readRDS("test_data_competing.rds")
natural_course <- natural_course(data = data, treat_var = "A")
natural_course
```

`plot_risk` *Plot risk estimated by ICE estimator over time*

Description

This function provides visualization of estimated risk on all user-defined interventions from the fitted ICE estimator object with natural course risk over time. (This function is going to be converted to the S3 method in R so it is named "plot_risk" for now. After conversion, users could use the base function plot().)

Usage

```
plot_risk(..., plot_np = T, label = 0)
```

Arguments

...	ICE estimator objects.
plot_np	a logical indicating whether to add the nonparametric natural course risk over time to the plot. Default is TRUE
label	a numeric specifying which time label is used in x axis. 0 represents using generic numerical time index, and 1 represents using the original time index in the data set. Default is 0.

Value

a plot for risks of all the interventions specified in ....

Examples

```
fit_classical_pool <- ice(
data = data,
time_points = 4,
id = "id",
time_name = "t0",
outcome_name = "Y",
censor_name = "C",
compevent_name = "D",
estimator = pool(hazard = F),
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
int_descript = c("Static Intervention"),
intervention1.A1 = list(static_A1),
intervention1.A2 = list(static_A2)
)

fit_hazard_pool <- ice(
data = data,
time_points = 4,
id = "id",
time_name = "t0",
outcome_name = "Y",
censor_name = "C",
compevent_name = "D",
estimator = pool(hazard = T),
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
int_descript = c("Static Intervention"),
intervention1.A1 = list(static_A1),
intervention1.A2 = list(static_A2)
)

plot_risk(fit_classical_pool, fit_hazard_pool)
```

pool *Indicator for the pooling over treatment history ICE estimator in the main function* ice

Description

This function identifies the pooling over treatment history ICE estimator. The classical pooling over treatment history ICE estimator is specified by `hazard = F`. The hazard based pooling over treatment history ICE estimator is specified by `hazard = T`.

Usage

```
pool(hazard)
```

Arguments

hazard	a logical indicating whether to use hazard-based ICE estimator or classical ICE estimator. TRUE for hazard-based estimator. FALSE for classical estimator.

Value

a logical on whether to use hazard-based ICE estimator.

static *Static Intervention Function*

Description

This function implements the static intervention, specifically always treat and never treat. Always treat refers to the constant treatment across all time points. Never treat refers to no treatment across all time points.

Usage

```
static(value, data = interv_data)
```

Arguments

value	a numeric specifying the intervention value. 1 for always treat and 0 for never treat.
data	a data frame containing the observed data.

Value

a vector containing the intervened value in the same size as the number of rows in `data`.

Examples

```
data <- readRDS("test_data_competing.rds")
always_treat <- static(value = 1, data = data)
always_treat
```

strat — *Indicator for the stratifying treatment history ICE estimator in the main function* ice

Description

This function identifies the stratifying treatment history ICE estimator. The classical stratifying treatment history ICE estimator is specified by `hazard = F`. The hazard based stratifying treatment history ICE estimator is specified by `hazard = T`.

Usage

```
strat(hazard)
```

Arguments

hazard	a logical indicating whether to use hazard-based ICE estimator or classical ICE estimator. TRUE for hazard-based estimator. FALSE for classical estimator.

Value

a logical on whether to use hazard-based ICE estimator.

summary_table — *Output the Summary Table from ICE Estimator Object*

Description

(This function is going to be converted to the S3 method in R so it is named "summary_table" for now. After conversion, users could use the base function summary().)

Usage

```
summary_table(...)
```

Arguments

...	the ICE estimator objects.

Value

a data frame containing the summary table for all specified ICE estimator objects.

Examples

```
fit_classical_pool <- ice(
data = data,
K = 4,
id = "id",
time_name = "t0",
outcome_name = "Y",
censor_name = "C",
competing_name = "D",
estimator = pool(hazard = F),
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
int_descript = c("Static Intervention"),
intervention1.A1 = list(static_A1),
intervention1.A2 = list(static_A2)
)

fit_hazard_pool <- ice(
data = data,
K = 4,
id = "id",
time_name = "t0",
outcome_name = "Y",
censor_name = "C",
competing_name = "D",
estimator = pool(hazard = T),
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
int_descript = c("Static Intervention"),
intervention1.A1 = list(static_A1),
intervention1.A2 = list(static_A2)
)

summary_table(fit_classical_pool, fit_hazard_pool)
```

---

threshold *Threshold Intervention*

Description

This function implements the threshold intervention. At each time, if an individual's treatment value is within the user-specified range inclusively, then follow the natural value of the treatment. Otherwise, if the treatment value is below the lower bound specified by the user, then set to the lower bound of the range, and if the treatment value is above the upper bound specified by the user, then set to the upper bound of the range. For more details, please see Young et al 2014.

Usage

```
threshold(
  lower_bound,
```

```
  upper_bound,
  var = threshold_treatment,
  data = interv_data
)
```

Arguments

lower_bound	a numeric specifying the lower bound of the threshold.
upper_bound	a numeric specifying the upper bound of the threshold.
var	a character string specifying the treatment variable for the intervention.
data	a data frame containing the observed data.

Value

the intervened values on the intervention variable.

References

Young JG, Herńan MA, Robins JM. Identification, estimation and approximation of risk under interventions that depend on the natural value of treatment using observational data. Epidemiologic Methods. 2014;3(1):1-19.

Examples

```
data <- readRDS("test_data_competing.rds")
threshold_treat <- threshold(threshold = -3, var = "A", data = data)
threshold_treat
```

---

weight *Indicator for the weighted ICE estimator in the main function* ice

---

Description

This function identifies the doubly robust weighted ICE estimator. The treatment models could be specified by treat_model and the treatment variables could be specified by obs_treatment_varnames.

Usage

```
weight(treat_model = list())
```

Arguments

treat_model	a list of formulas specifying the model statement for the corresponding treatment variable used in the doubly robust ICE estimator. The length of list must match with the length of obs_treatment_varnames. Multiple treatments passed in obs_treatment_varnames are allowed and must follow the corresponding order as in treat_model_covar.
obs_treatment_varnames	a list of character strings specifying the treatment variables to be used in the model for observed treatments of the weighted ICE estimator.

Value

the model specification for each treatment model and the treatment variables and the observed treatment variable names extracted from the specified model.

Index